```
 1: unit LstEnc;
 2:
 3: interface
 4:
 5: type
 6:    { Tipo de chave do item da lista }
 7:    Tipo_Chave = integer;
 8:
 9:    { Tipo do item }
10:    Tipo_Item = record
11:                   Chave   : Tipo_Chave;
12:                   Dado    : String[30];
13:                end;
14:
15:    { Tipo do apontador para um item da lista }
16:    Pont_Item_Lista = ^Item_Lista;
17:
18:    { Tipo do item da lista }
19:    Item_Lista = record
20:                   Item : Tipo_Item;
21:                   Proximo : Pont_Item_Lista
22:                 end;
23:
24:    { Tipo da lista Encadeada }
25:    Lista_Encadeada = record
26:                         Cabeca : Pont_Item_Lista;
27:                       end;
28:
29: procedure Inicializar (var Lista : Lista_Encadeada);
30: function Inserir (Item : Tipo_Item; var Lista : Lista_Encadeada) : boolean;
31: function Remover (Chave : Tipo_Chave; var Lista : Lista_Encadeada) : boolean;
32: function Alterar (Item : Tipo_Item; var Lista : Lista_Encadeada) : boolean;
33: procedure Obter (Chave : Tipo_Chave; var Lista : Lista_Encadeada;
34:                  var Item : Tipo_Item; var Sucesso : boolean);
35: procedure Apagar (var Lista : Lista_Encadeada);
36: function Tamanho (var Lista : Lista_Encadeada) : integer;
37: function Cheia (var Lista : Lista_Encadeada) : boolean;
38: function Vazia (var Lista : Lista_Encadeada) : boolean;
39:
40: implementation
41:
42: procedure Inicializar (var Lista : Lista_Encadeada);
43: {
44:   Objetivo: Inicializa a lista passada, fazendo com que a cabeca da lista
45:             aponte para nil.
46: }
47: begin
48:    Lista.Cabeca := nil
49: end;
50:
51:
52: function Inserir (Item : Tipo_Item; var Lista : Lista_Encadeada) : boolean;
53: {
54:   Objetivo: Insere o item passado como parametro na lista passada.
55:             Se a lista ja estiver cheia, a funcao Inserir retorna false.
56: }
57: var
58:    PNovo : Pont_Item_Lista;
59: begin
60:    if Cheia(Lista) then
61:       Inserir := false
```

```
62:    else
63:       begin
64:          New(PNovo);
65:          PNovo^.Item := Item;
66:          PNovo^.Proximo := Lista.Cabeca;
67:          Lista.Cabeca := PNovo;
68:          Inserir := true
69:       end
70: end;
71:
72:
73: function Remover (Chave : Tipo_Chave; var Lista : Lista_Encadeada) : boolean;
74: {
75:   Objetivo: Remove o item cuja chave coincide com o parametro Chave
76:             passado. Caso nao haja um item com essa chave, retorna
77:             false. Se o item foi removido, retorna true.
78: }
79: var
80:    PAtual, PAnterior : Pont_Item_Lista;
81: begin
82:    Remover := false;
83:
84:    if Vazia(Lista) then
85:       exit;
86:
87:    PAnterior := Lista.Cabeca;
88:
89:    if Lista.Cabeca^.Item.Chave = Chave then
90:       begin
91:          { Remove o item da cabeca da lista e a cabeca passa a ser
92:            o item seguinte ao removido}
93:          Lista.Cabeca := Lista.Cabeca^.Proximo;
94:          dispose(PAnterior);
95:          Remover := true
96:       end
97:    else
98:       { Percorre a lista ate encontrar um item com a chave procurada.
99:         Remove o item e corrige o apontador do item anterior para
100:        apontar para o proximo item }
101:      while PAnterior^.Proximo <> nil do
102:      begin
103:         PAtual := PAnterior^.Proximo;
104:
105:         if PAtual^.Item.Chave = Chave then
106:            begin
107:               PAnterior^.Proximo := PAtual^.Proximo;
108:               Dispose(PAtual);
109:               Remover := true;
110:               break
111:            end;
112:
113:         PAnterior := PAtual;
114:      end
115: end;
116:
117:
118: function Alterar (Item : Tipo_Item; var Lista : Lista_Encadeada) : boolean;
119: {
120:   Objetivo: Altera os dados de um item existente na lista passada
121:             de forma que fique igual ao do item passado como parametro.
122:             Se o item for encontrado e alterado, retorna true. Caso
```

```
123:             contrario, retorna false.
124: }
125: var
126:    PAtual : Pont_Item_Lista;
127: begin
128:    Alterar := false;
129:    PAtual := Lista.Cabeca;
130:
131:    while PAtual <> nil do
132:       if PAtual^.Item.Chave = Item.Chave then
133:          begin
134:             PAtual^.Item := Item;
135:             Alterar := true;
136:             break
137:          end
138:       else
139:          PAtual := PAtual^.Proximo
140: end;
141:
142:
143: procedure Obter (Chave : Tipo_Chave; var Lista : Lista_Encadeada;
144:                  var Item : Tipo_Item; var Sucesso : boolean);
145: {
146:   Objetivo: Procura na lista usando a chave passada. Caso encontre
147:             Sucesso contem o valor true e Item contem o Item obtido.
148:             Caso contrario, Sucesso retorna true e Item nao e alterado
149: }
150: var
151:    PAtual : Pont_Item_Lista;
152: begin
153:    Sucesso := false;
154:    PAtual := Lista.Cabeca;
155:
156:    while PAtual <> nil do
157:       if PAtual^.Item.Chave = Chave then
158:          begin
159:             Item := PAtual^.Item;
160:             Sucesso := true;
161:             break
162:          end
163:       else
164:          PAtual := PAtual^.Proximo
165: end;
166:
167:
168: procedure Apagar (var Lista : Lista_Encadeada);
169: {
170:   Objetivo: Apaga a lista passada
171: }
172: var
173:    PAtual, PApagar : Pont_Item_Lista;
174: begin
175:    PAtual := Lista.Cabeca;
176:
177:    while PAtual <> nil do
178:    begin
179:       PApagar := PAtual;
180:       PAtual := PAtual^.Proximo;
181:       Dispose(PApagar)
182:    end;
183:
```

```
184:    Lista.Cabeca := nil
185: end;
186:
187:
188: function Tamanho (var Lista : Lista_Encadeada) : integer;
189: {
190:   Objetivo: Retorna o tamanho da lista passada
191: }
192: var
193:    P : Pont_Item_Lista;
194:    Cont : integer;
195: begin
196:    P := Lista.Cabeca;
197:    Cont := 0;
198:
199:    while P <> nil do
200:    begin
201:       inc(Cont);
202:       P := P^.Proximo
203:    end;
204:
205:    Tamanho := Cont;
206: end;
207:
208:
209: function Cheia (var Lista : Lista_Encadeada) : boolean;
210: {
211:   Objetivo: Retorna true se nao ha mais memoria disponivel
212:             para inserir um item na lista
213: }
214: begin
215:    Cheia := MaxAvail < SizeOf(Item_Lista)
216: end;
217:
218:
219: function Vazia (var Lista : Lista_Encadeada) : boolean;
220: {
221:   Objetivo: Retorna true se a lista esta vazia (Cabeca = nil)
222: }
223: begin
224:    Vazia := Lista.Cabeca = nil
225: end;
226:
227: end.
```